Swap Solutions

Copyright © James Raynard 2021

Swapping

- Write a program which creates two std::string objects and swaps their data using
  - A member function of std::string
  - A non-member function from the standard library

Copyright © James Raynard 2021

Implementation of swap()

- What performance issues can arise when swapping two complex objects?
  - The direct approach of creating a temporary object and assigning to it can have a lot of overhead
- How does the standard library address this?
  - Instead of creating a new object, it swaps the data members of the existing objects
  - This is done by implementing overloads of std::swap() for every container in the STL
  - There is also a generic std::swap() that uses copying
  - All STL algorithms which exchange container elements call swap()

Copyright © James Raynard 2021

Should we overload swap?

- Is it worthwhile overloading swap() for our own classes?
  - Yes, if objects of the class are expensive to copy and are likely to be passed to swap()
  - For example, if there is a container of these objects that is sorted by calling std::sort()
- What happens if we do not overload swap()?
  - The generic std::swap() will be called
  - This will use copying to perform the swap operation
  - This could impair performance

Copyright © James Raynard 2021

Implementing swap

- What issues should we consider when overloading swap() for our class?
  - The swap() implementation should be very fast and not throw exceptions
  - It must be a non-member function which is either a friend of the class, or calls a public member function of the class which performs the actual swap operation
  - It should be declared inline
  - It should be implemented by calling the overload of swap() for each member
  - This can be done by putting the following statement at the start of the function

```
using std::swap;
```

Copyright © James Raynard 2021

swap() overload example

- Write a simple class
- Implement an overload of swap() for your class
- Write a program to test your code

Copyright © James Raynard 2021